Errors

B. Allombert

Institut de Mathématiques de Bordeaux
Univ. Bordeaux, CNRS, INRIA

24/06/2025

printf

- `void pari_printf(const char *fmt, ...)` is a variant of `printf` which supports the size modifier `P` for `GEN` object.

```
pari_printf("%d^%d = %Pd\n",2, 128, shifti(2,128));
pari_printf("M = %Ps\n", mathilbert(5));
pari_printf("Pi = %Pg\n", mppi(128));
```

- `void err_printf(const char *fmt, ...)` output to the error channel.
- `GEN gsprintf(const char *fmt, ...)` return the string as a `t_STR`.
- `char * pari_sprintf(const char *fmt, ...)` return the string as a `char *` that needs to be freed with `pari_free`.
- `char * stack_sprintf(const char *fmt, ...)` return the string as a `char *` on the PARI stack.

timer

PARI timers are of type `pari_timer`.

- `void timer_start(pari_timer *ti)`: start the timer.
- `long timer_delay(pari_timer *ti)`: return the elapsed time in millisecond.
- `void timer_printf(pari_timer *ti, const char *format, ...)` print `Time str :  delay` where `str` is `stack_sprintf(const char *fmt, ...)` and `delay` is the time in millisecond. This function also resets the timer.

Debug domain

Debug domains set by `setdebug` are defined in `src/headers/paridbglvl.h`. A C file is associated with a debug domain by adding at the start of the file the line (here for the domain `bnf`)

```
#define DEBUGLEVEL DEBUGLEVEL_bnf
```

then when one want to indicate progress, one do

```
if (DEBUGLEVEL>=1) err_printf(...);
```

or

```
if (DEBUGLEVEL>=1) timer_printf(&ti,...);
```

Example

```
GEN myfun(GEN V)
{
  pari_timer ti;
  long i, l = lg(V);
  GEN W = cgetg(l, t_VEC);
  timer_start(&ti);
  for (i=1; i<l; i++)
  {
    gel(W, i) =  Z_factor(gel(V,i));
    if (DEBUGLEVEL>=1)
      timer_printf(&ti,"factoring entry %ld",i);
  }
  return W;
}
```

Throwing errors

Errors can be thrown using `pari_err_xxx` functions (that in turn call the internal `pari_err` function). The data provided is encapsulated in a `t_ERROR` object returned by `iferr`.

- `void pari_err_BUG(const char *f)`, message `"bug in f, please report."`.
- `void pari_err_TYPE(const char *f, GEN x)`, message `"incorrect type in f (type(x))`.
- `void pari_err_TYPE2(const char *f, GEN x, GEN y)`, message `"forbidden f type(x), type(y).`
- `void pari_err_DOMAIN(const char *f, const char *v, const char *op, GEN l, GEN x)`, message `"domain error in f:  v op x."`
- `void pari_err_INV(const char *f, GEN x)`, message `"impossible inverse in f:  x."`

Example GP

```
? install(pari_err_DOMAIN,"vsssGG","err");
? err("ellj","Im(tau)","<=",gen_0,-I)
  *** err: domain error in ellj: Im(tau) <= gen_0
? E=iferr(err("ellj","Im(tau)","<=",gen_0,-I),E,E);
%3 = error("domain error in ellj: Im(tau) <= gen_0")
? Vec(E)
%4 = ["e_DOMAIN","ellj","Im(tau)","<=",gen_0,-I]
```

Example C

```
GEN modii_chk(GEN x, GEN y}
{
  if (typ(x)!=t_INT) pari_err_TYPE("modii_chk", x);
  if (typ(y)!=t_INT) pari_err_TYPE("modii_chk", y);
  if (signe(y)==0)   pari_err_INV("modii_chk", y);
  return modii(x,y);
}
```

t_ERROR

t_ERROR denote error objects returned by iferr.

- long err_get_num(GEN E): return the type of the error (e_BUG, e_TYPE, e_DOMAIN, e_INV, etc.).
- GEN err_get_compo(GEN E, long i): return the i-th component of the error.

An universal t_ERROR object err_e_STACK is available, which is safe to use even after a PARI stack overflow (e_STACK or e_STACKTHREAD).

Catching errors

Catch errors in C is discouraged but possible (at least to implement `iferr`).

- `pari_CATCH(e_TYPE)` catch errors of type `e_TYPE`, use `CATCH_ALL` to catch all errors (very discouraged). Introduce the error recovery code. It is possible to recover from a stack overflow by resetting `avma` to the value just before `pari_CATCH`.
- `GEN pari_err_last(void)`: return the error that was caught as a `t_ERROR`.
- `pari_TRY`: Introduce the code to execute.
- `pari_ENDCATCH`: End of the code to execute.

Examples

```
GEN ecm(GEN N, GEN B, GEN nb)
{ long a;
  pari_CATCH(e_INV)
  {
    GEN E = pari_err_last();
    return ggcd(lift(err_get_compo(E, 2)), N);
  } pari_TRY {
    for (a = 1; a<=nb; a++)
    {
      GEN E = ellinit(mkvec2s(a,1), N, DEFAULTPREC);
      ellmul(E, mkvec2s(0,1), B);
    }
  } pari_ENDCATCH
  return gen_0;
}
```